GREW, un outil au service de l'annotation de corpus et de l'exploitation de corpus annotés

Guy Perrier, Bruno Guillaume

GREW, un outil au service de l'annotation de corpus et de l'exploitation de corpus annotés

Guy Perrier[1] Bruno Guillaume[1]
(1) LORIA, 54506 Vandœuvre-lès-Nancy cedex, France
guy.perrier@loria.fr, bruno.guillaume@loria.fr

RÉSUMÉ

Nous présentons GREW, un outil dédié à l'annotation de corpus et à l'exploitation de corpus annotés. Fondé sur la réécriture de graphes, il peut être utilisé d'une part pour rechercher des motifs dans des données annotées et d'autre part pour transformer ce type de données.

ABSTRACT

Graph rewriting for corpus annotation and annotated corpus exploitation

We present GREW, a tool dedicated to corpus annotation and the exploitation of annotated corpora. Based on graph rewriting, it can be used to search for patterns in annotated data and to transform this type of data.

MOTS-CLÉS : réécriture de graphes, annotation de corpus.

KEYWORDS: graph rewriting, corpus annotation.

Que ce soit pour la linguistique de corpus ou le TAL fondé sur de l'apprentissage profond ou pas, il est de plus en plus nécessaire de disposer de corpus annotés de qualité et de taille importante, ainsi que d'outils automatiques pour les manipuler. De quelles manipulations, a-t-on le plus souvent besoin ? De pouvoir retrouver un motif dans une annotation, de corriger une annotation, de pouvoir la convertir d'un format dans un autre à un même niveau linguistique, ou encore de produire une annotation à un niveau à partir d'une annotation à un autre niveau. Le développement d'outils automatiques dédiés à ces différentes tâches est facilité par l'inscription de ceux-ci dans un cadre mathématique commun.

1 Le choix du cadre mathématique des graphes

Les graphes s'imposent naturellement pour la représentation des structures sémantiques, ne serait-ce que parce qu'une entité peut être en même temps l'argument de plusieurs prédicats. Les structures syntaxiques, quant à elles, sont en général des arbres, c'est-à-dire des graphes particuliers. Or qui peut le plus, peut le moins. Par ailleurs, il est parfois utile de considérer l'interaction du niveau syntaxique avec un niveau linguistique voisin. Si par exemple, on veut représenter en même temps les relations syntaxiques et les relations d'ordre entre les mots, les structures qui en résultent sont des graphes.

Tout en restant dans le cadre de la syntaxe, plusieurs théories linguistiques (Sgall et al., 1986; Mel'čuk, 1988) ont conçu un niveau intermédiaire (appelé syntaxe profonde) entre la syntaxe (rebaptisée syntaxe de surface) et la sémantique. En syntaxe profonde, il s'agit de représenter uniquement les relations entre mots lexicaux, même quand elles sont indirectes ou implicites. Les structures qui en résultent

sont en général des graphes.

L'outil que nous avons conçu avec Guillaume Bonfante, GREW [1], est fondé sur la réécriture de graphes (Bonfante et al., 2018). Un système de réécriture de graphes est un ensemble de règles qui décrivent des transformations élémentaires et qui sont appliquées successivement pour réaliser une transformation plus globale. Chaque règle est formée d'une partie gauche qui décrit un motif qui doit être remplacé dans un graphe et d'une partie droite qui indique par quoi ce motif doit être remplacé. GREW est utilisé de deux façons en TAL : pour rechercher un motif dans une annotation et pour transformer une annotation.

2 La recherche de motifs dans un corpus annoté

Il est très utile de pouvoir retrouver un motif donné dans un corpus annoté, que ce soit à des fins linguistiques ou pour détecter des erreurs et des incohérences de manière à les corriger.

Si une annotation se présente comme un graphe, et si on considère aussi un motif comme un graphe, le problème revient à apparier deux graphes. Le module de GREW dédié à cette tâche, GREW-MATCH, fournit une syntaxe très simple pour décrire les motifs à rechercher, aussi complexes soient-ils.

Supposons que nous voulions étudier dépendances à distance dans les propositions relatives dans un corpus annoté en syntaxe de dépendances selon le format de Universal Dependencies (UD) [2]. Le chemin de dépendances qui va de l'antécédent du pronom relatif jusqu'au pronom lui-même, en passant par la tête de la relative peut être plus ou moins long.

Le motif suivant, écrit dans la syntaxe de GREW, permet déjà de lister toutes les constructions avec subordonnées relatives :

```
pattern { ANT -[acl:relcl]-> REL_HEAD }
```

Les identifiants `ANT` et `REL_HEAD` sont utilisés pour nommer les nœuds source (l'antécédent d'un pronom relatif) et cible (la tête de la relative) pour pouvoir y faire référence par la suite. L'étiquette `acl:relcl` exprime une dépendance de l'antécédent vers la tête d'une relative.

GREW permet d'ajouter au motif des conditions négatives pour filtrer des solutions que l'on souhaite écarter. Par exemple, enrichissons le motif précédent avec une condition négative :

```
pattern { ANT -[acl:relcl]-> REL_HEAD }
without { PROREL [cat=PRON, Prontype=Rel]; REL_HEAD -> PROREL }
```

La condition négative introduite par le mot-clé `without` permet d'écarter tous les cas où le pronom relatif `PROREL` est rattaché directement à la tête `REL_HEAD` de la relative, et ainsi de lister les cas où le pronom relatif est enchâssé.

Par l'ajout successif de conditions négatives, on peut affiner progressivement la recherche. Par exemple, on peut vouloir écarter de l'étude les cas où le pronom relatif est enchâssé dans un groupe prépositionnel. Il suffit d'ajouter au motif la condition négative suivante.

1. http://grew.fr/
2. https://universaldependencies.org/

```
without {
  REL_HEAD -> PREP; PREP [cat=P];
  PREP -[obj.p]-> PROREL; PROREL [cat=PRO, s=rel];}
```

Cette façon de procéder est très générale et sa mise œuvre est facile car l'outil est disponible en ligne sans installation [3].

3 La transformation d'annotations par réécriture de graphes

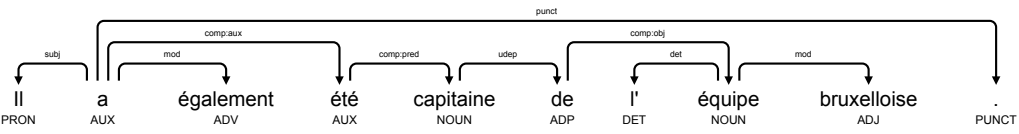
La réécriture de graphes est un prolongement de l'opération d'appariement présentée dans la section précédente. Un système de réécriture de graphes est un ensemble de règles locales de transformations élémentaires de graphe. Une règle est formée d'une partie gauche (qui décrit le motif à rechercher dans le graphe) et d'une partie droite (qui décrit comment modifier le graphe). La difficulté est de préciser comment le résultat de l'application de la règle va être connecté au contexte. Il n'y a pas mathématiquement de façon standard de faire, nous avons donc conçu un modèle de la réécriture de graphes spécifiquement adapté au TAL, où la partie droite des règles se présente comme une suite d'opérations élémentaires réalisant la modification du graphe (ajout ou suppression de nœuds ou d'arcs par exemple).

En pratique, de nombreuses applications de règles sont nécessaires pour effectuer une transformation et il est crucial de contrôler leur enchaînement, ce que permet GREW avec la notion de *stratégie*. Une stratégie est un moyen de décrire l'ordre selon lequel des règles ou des paquets de règles s'enchaînent.

De quelles transformations a-t-on besoin quand on travaille avec des annotations de corpus ? Ce peut être tout d'abord des corrections d'erreurs systématiques d'annotation. Ce peut être aussi produire une annotation à un niveau linguistique donné à partir d'un niveau voisin, par exemple produire une annotation sémantique à partir d'une annotation syntaxique. Ce peut être enfin convertir une annotation d'un format dans un autre format en restant au même niveau linguistique.

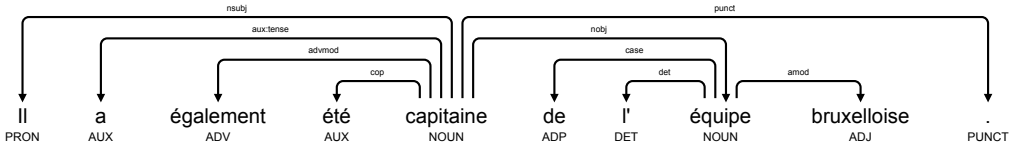
Par exemple, nous avons écrit un système de règles pour convertir une annotation syntaxique du format Universal Dependencies (UD) au format Surface Syntactic Universal Dependencies (SUD) [4] (Gerdes et al., 2019) et un autre système pour faire la conversion inverse.

Voici un exemple de phrase annoté dans le format SUD.



Voici maintenant la même phrase annotée dans le format UD, l'annotation ayant été obtenu par application de notre système de règles de réécriture.

3. http://match.grew.fr
4. https://surfacesyntacticud.github.io/



Pour obtenir cette annotation, il a fallu appliquer 24 règles. Une différence importante entre les deux formats, que l'on peut remarquer sur l'exemple ci-dessus, est le choix des têtes des dépendances : pour les expressions introduites par des mots fonctionnels (auxiliaires, prépositions, conjonctions), dans SUD, les têtes sont ces mots fonctionnels alors que dans UD, les têtes sont les mots lexicaux. Voici la règle, un peu simplifiée, qui a effectué le changement de tête des relations *aux :tense* et *cop* [5].

```
rule rev_head {
  pattern {e:HEAD -[aux:tense|cop]-> AUX;}
  without {HEAD[reversed=y]; AUX[reversed=y] }
  commands {
    HEAD.reversed=y; AUX.reversed=y;
    add_edge e:AUX -> HEAD; del_edge e;
    shift_in HEAD ==> AUX;
    shift_out HEAD =[^unk:fixed]=> AUX}}
```

Elle est formée de deux parties :

— La première partie, avec une clause `pattern` et une clause `without`, décrit le motif à rechercher dans l'annotation. La clause `without` permet de s'assurer que la dépendance n'a pas déjà été retournée. Le fonctionnement de la recherche du motif est identique à ce qu'on a vu dans la section précédente.

— La partie `commands` présente la suite de commandes qui décrivent les modifications souhaitées sur le graphe. Les deux premières commandes consistent à marquer les nœuds `HEAD` et `AUX` du trait `reversed=y` pour indiquer que le retournement est effectué. Les deux commandes suivantes effectuent le retournement. La commande `shift_in` déplace toutes les relations qui arrivent de l'ancienne tête vers la nouvelle et la commande `shift_out` fait de même pour les relations qui partent à l'exception de celles étiquetées `unk:fixed`.

Références

Bonfante, G., Guillaume, B., and Perrier, G. (2018). *Application de la réécriture de graphes au traitement automatique des langues*, volume 1 of *Série Logique, linguistique et informatique*. ISTE editions.

Gerdes, K., Guillaume, B., Kahane, S., and Perrier, G. (2019). Improving Surface-syntactic Universal Dependencies (SUD) : surface-syntactic relations and deep syntactic features. In *TLT 2019, Treebanks and Linguistic Theories, Syntaxfest*, Paris, France.

Mel'čuk, I. (1988). *Dependency Syntax : Theory and Practice*. Albany, N.Y. : The SUNY Press.

Sgall, P., Hajicová, E., and Panevová, J. (1986). *The meaning of the sentence in its semantic and pragmatic aspects*. Springer Science & Business Media.

5. Cette règle s'applique une fois que les étiquettes de relations ont été changées, ce qui explique la présence des étiquettes `aux : tense` et `cop` dans la clause `pattern`.